tetrate

POWERING THE WORLD'S APPLICATION NETWORKS

Turning your cloud native apps inside out with a service mesh

Adam is a solutions engineering leader at Tetrate. Prior to Tetrate, Adam worked as a Field Principal for VMware's Modern Application Platform business unit. He has a passion for OSS software and has contributed to the Spring and Cloudfoundry projects. His focus for nearly the past decade has been helping Global 2000 companies modernize their infrastructure platforms and adopt cloud native application architectures. In the past Adam has presented on service mesh and modern application development at various conferences such as SpringOne and Nist DevSecOps.

Liam White

Staff Software Engineer, Tetrate

AGENDA

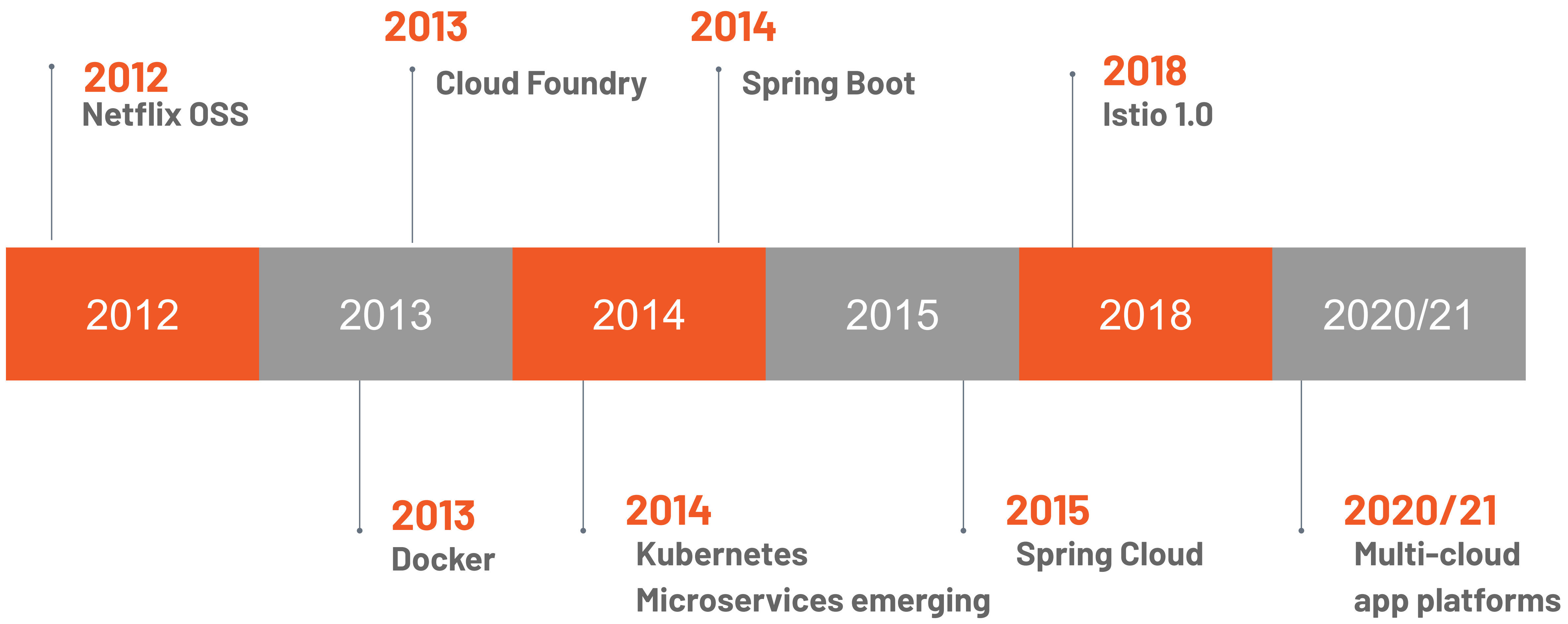
1. Building Cloud Native w/ Spring and Netflix
2. Where does Service Mesh fit in?
3. Migration Example

https://github.com/tetratelabs/tetrate-todos

tetrate

Adam & Tetrate

Modern Application Journey

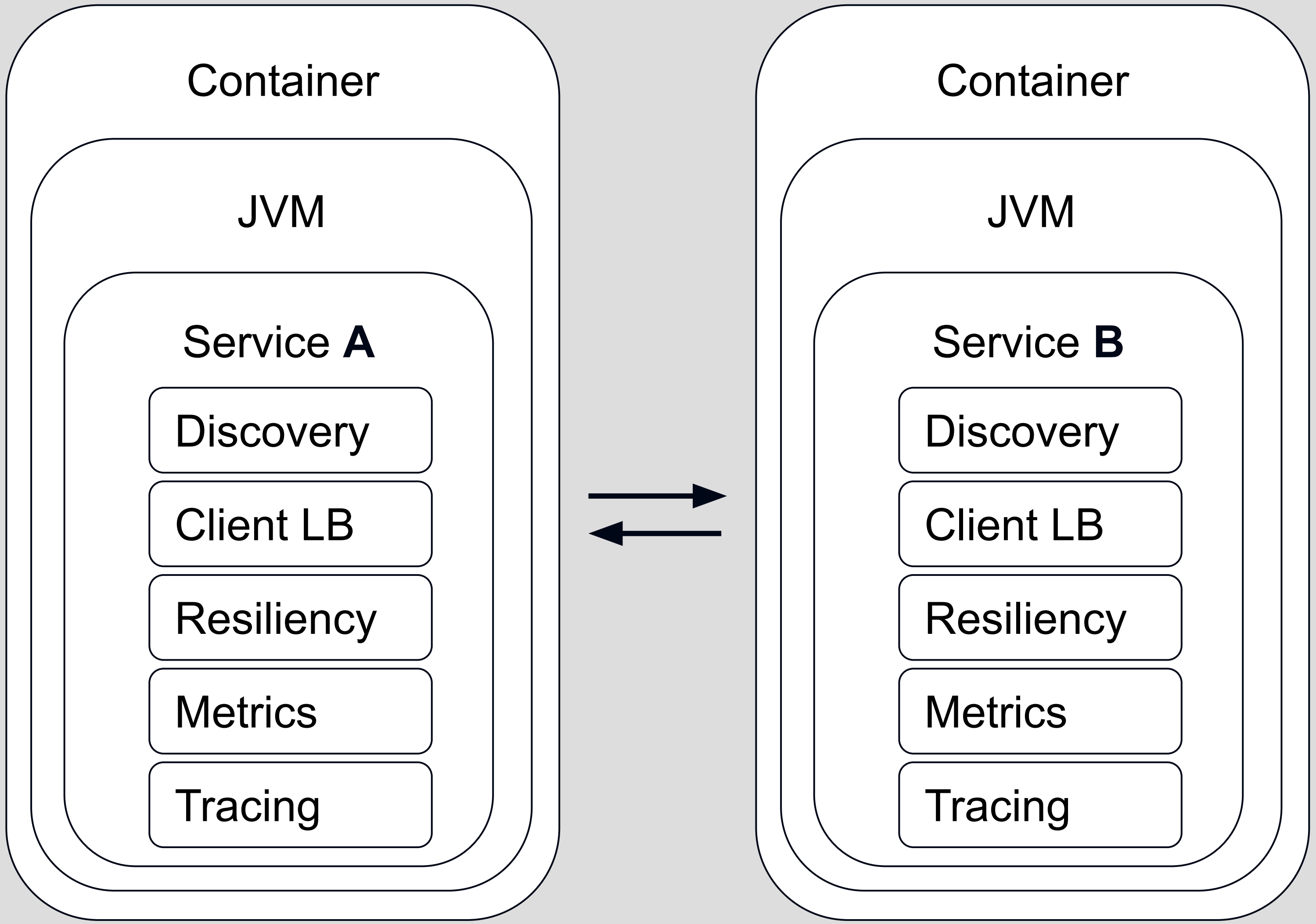


Ingredients to Cloud Native

All resources and dependencies are network attached

"Cloud native is a term describing software designed to run and scale reliably and predictably on top of potentially unreliable cloud-based infrastructure. "

- Duncan Winn



How???

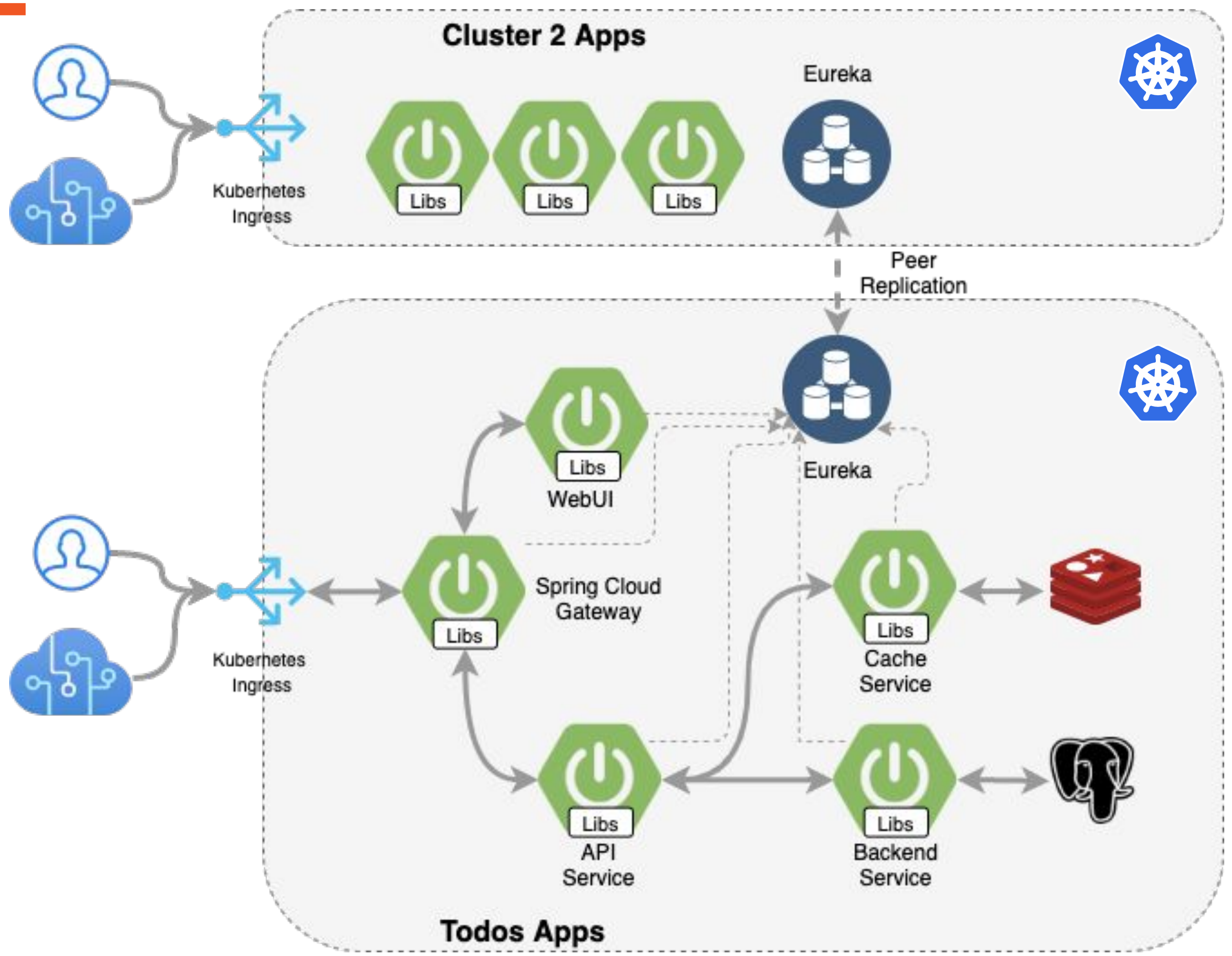
start.spring.io

```java
@SpringBootApplication
@EnableCircuitBreaker
public class TodosApi {
   public static void main(String[] args) { SpringApplication.run(TodosApi.class, args); }
}
....
   private String _cacheUrl = "http://todos-redis:8080";

   @LoadBalanced
   @Bean
   RestTemplate restTemplate(){ return new RestTemplate(); }

   @HystrixCommand(fallbackMethod = "failFastResult", commandProperties = {
       @HystrixProperty(name = "execution.isolation.thread.timeoutInMilliseconds", value = "1000"),
       @HystrixProperty(name = "circuitBreaker.requestVolumeThreshold", value = "10")})
   @GetMapping("/{id}")
   public Todo retrieve(@PathVariable("id") String id) {
       Todo cached = this.restTemplate().getForEntity(_cacheUrl + "/" + id, Todo.class).getBody();
```

application.yaml

```yaml
eureka:
 client:
   serviceUrl:
     defaultZone:
http://my-registry:8761/eureka/
 instance:
   preferIpAddress: true
```

Deployment Architecture



https://github.com/tetratelabs/tetrate-todos

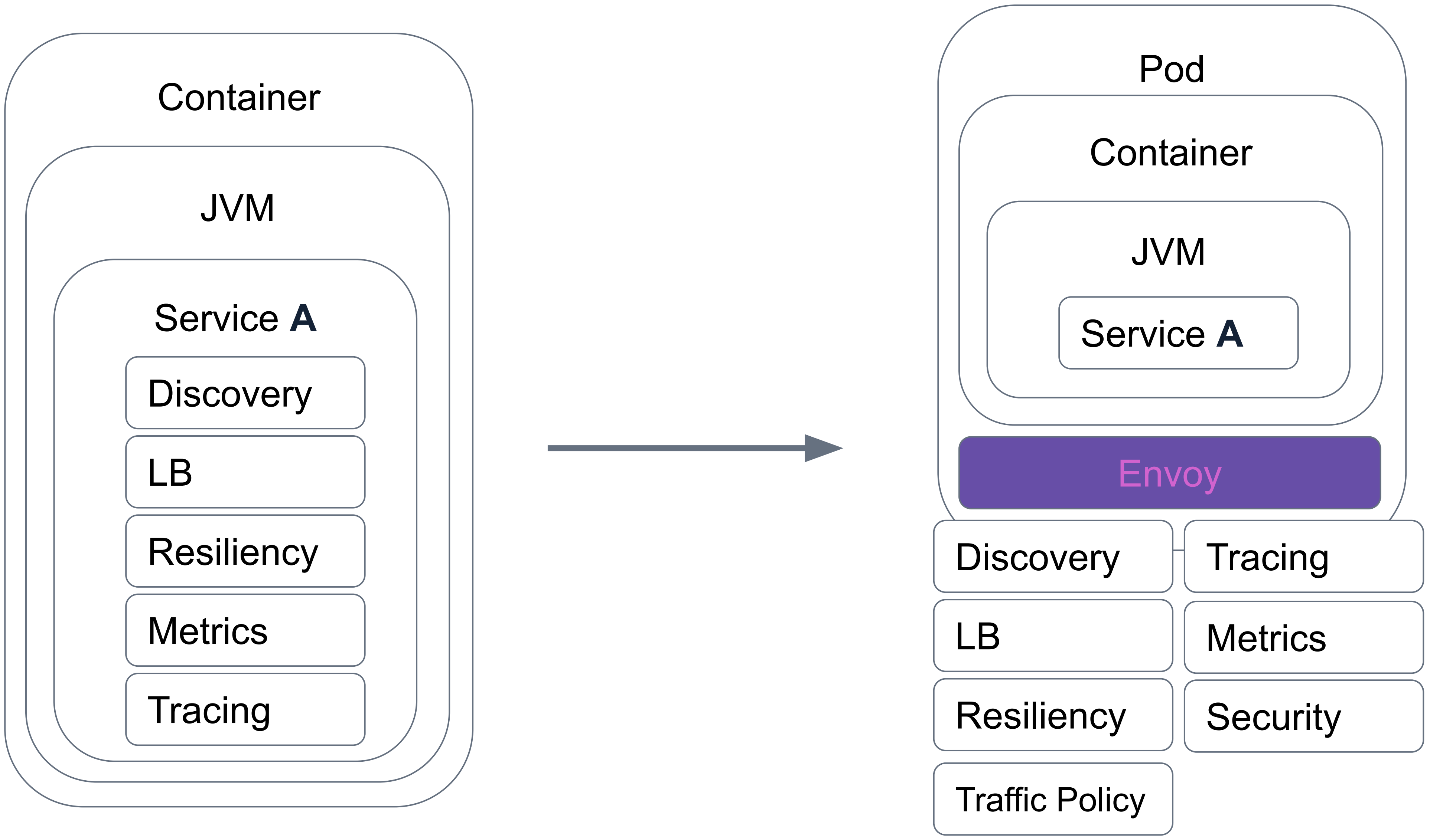
Gaps

- Polyglot experience
- Coupling with app (code, artifact, and CD)
- Overall Complexity
- Multi-cluster & multi-cloud semantics
- N/A for legacy or COTs
- Primitives for advanced security & CD

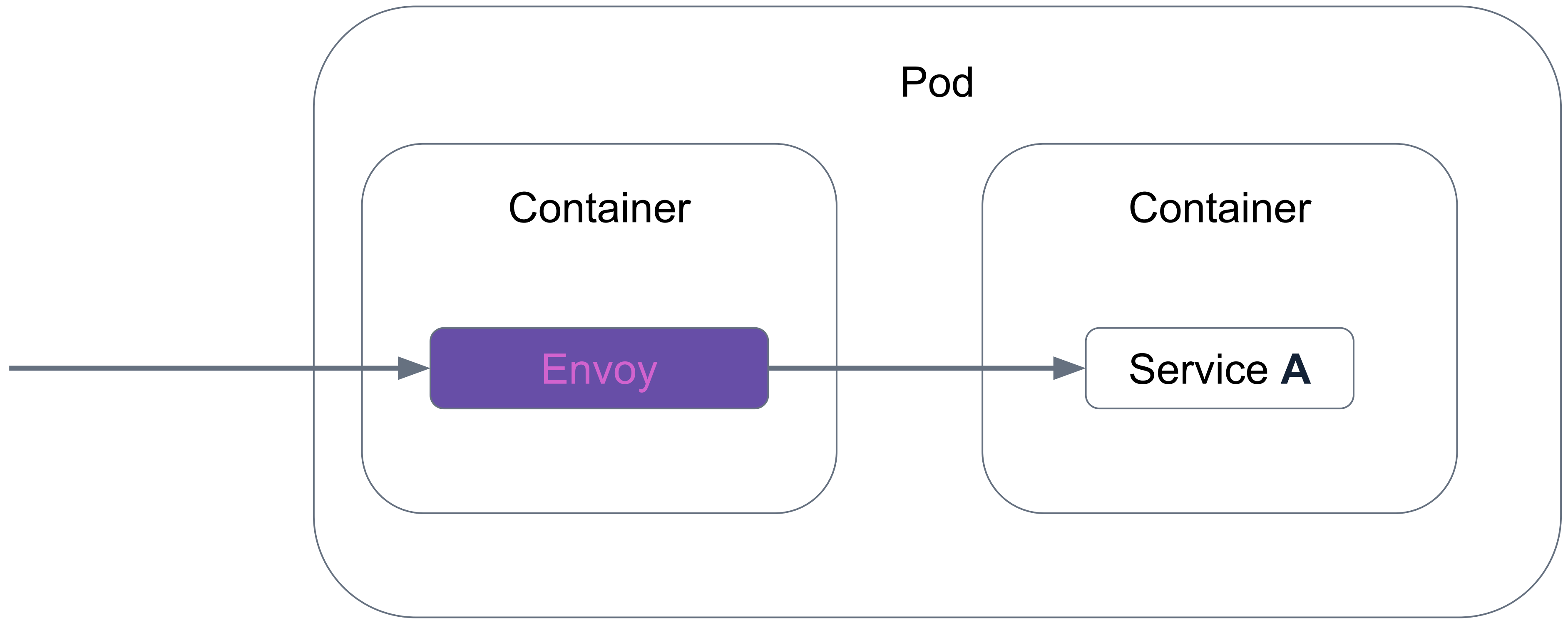
Transparent Network Layer

“The network should be transparent to applications. When network and application problems do occur it should be easy to determine the source of the problem.”

Netflix OSS -> Kubernetes + Envoy



Request Flow



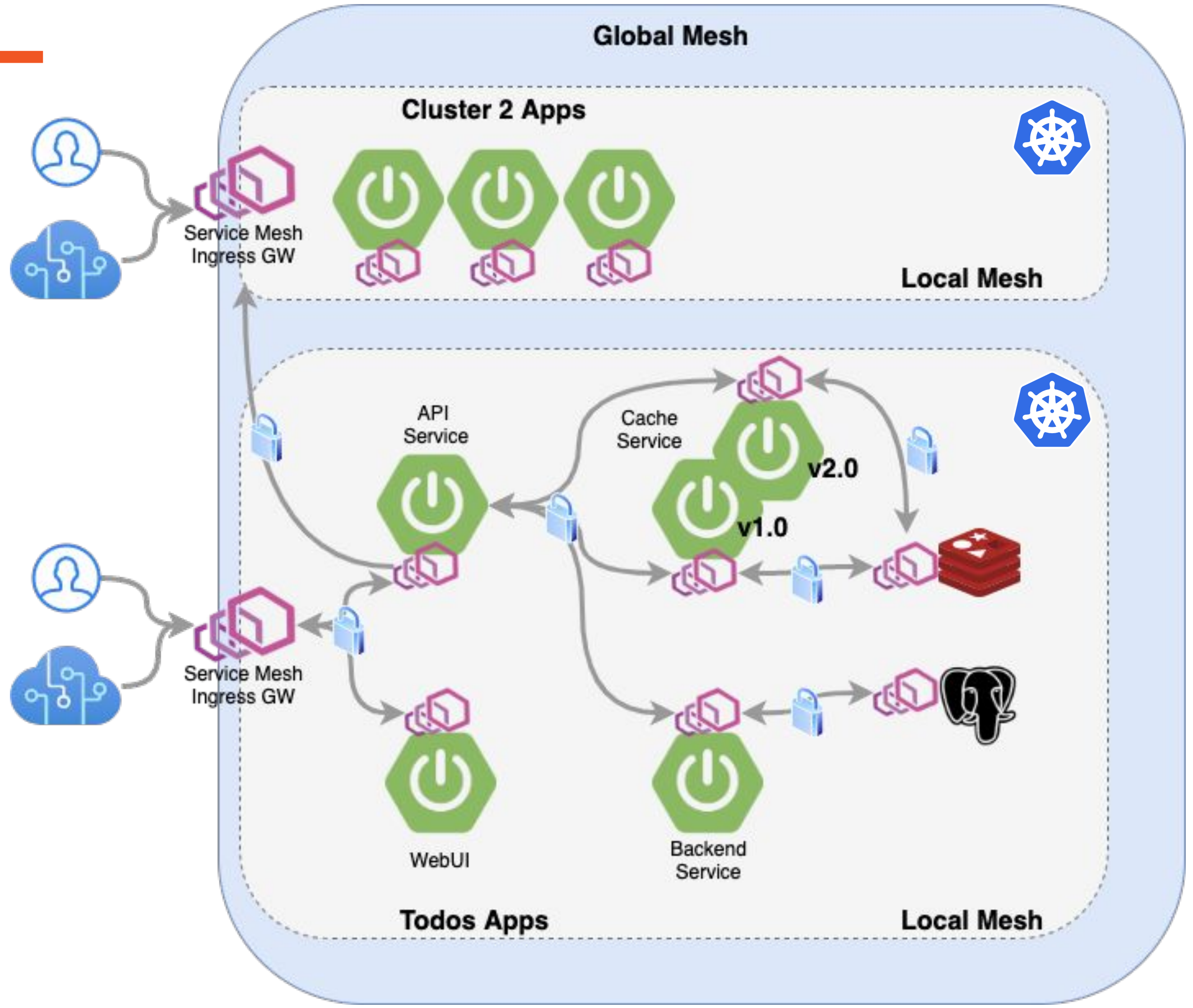
Out of Process Architecture

1. Works with any application language.
2. Works with any legacy and third party application.
3. Faster and transparent upgrades than libraries.
4. What about latency?

Consistency

1. Traffic Management configuration
2. Security/Policy configuration
3. Behavior
4. Telemetry

Deployment Architecture



https://github.com/tetratelabs/tetrate-todos

Mesh Capabilities

- Ingress & Service Discovery
- Client Side LB & Canaries
- Resiliency
- Security

Ingress and Service Discovery

App Changes

- **Optional** remove gateway
- Remove Eureka dependencies
- Remove @LoadBalanced, @DiscoveryClient, etc
- Retire Eureka Registry(s)

https://github.com/tetratelabs/tetrate-todos

1. Envoy utilized as Ingress Gateway

```
// A simple service definition
hosts:
- todos.tetrate.zwickey.net
gateways:
- todos-gw
http:
- match:
  - uri:
      prefix: /redis
  rewrite:
    uri: /
  route:
  - destination:
      host: todos-redis.todos.svc.cluster.local
      port:
        number: 8080
```

```
   private String _cacheUrl = "http://todos-redis:8080";

....

   @GetMapping("/{id}")

   public Todo retrieve(@PathVariable("id") String id) {

       Todo cached = restTemplate().getForEntity(_cacheUrl...
```

```
Envoy routes:
  - 8080 todos-redis /*

Envoy endpoints:
  - 10.16.3.17:8080     HEALTHY  OK  outbound|8080||todos-redis.todos.svc.cluster.local
  - 10.16.4.18:8080     HEALTHY  OK  outbound|8080||todos-redis.todos.svc.cluster.local
```

Client Side Load Balancing

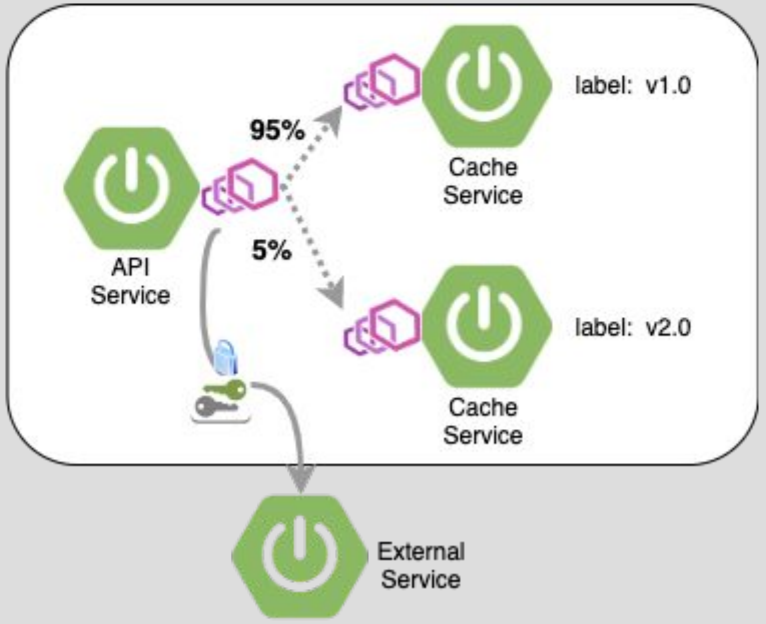
App Changes

- **Remove @LoadBalanced, @DiscoveryClient, etc**
- **Remove Ribbon or Load Balancer configs or implementations**

```
// A simple traffic splitting rule
hosts:
  - todos.tetrate.zwickey.net
http:
- route:
  - destination:
      host: todos-redis
      subset: v1.0
    weight: 95
    timeout: .5s
  - destination:
      host: todos-redis
      subset: v2.0
    weight: 5
    timeout: .5s
  retries:
      attempts: 3
      retryOn: connect-failure,gateway-error
```

```
// Add in LB Strategy
host: todos-redis
trafficPolicy:
  loadBalancer:
    simple: LEAST_CONN
```

```
// Connect externally w/ mTLS
host: external-service.tetrate.io
trafficPolicy:
  tls:
    mode: MUTUAL
    clientCertificate: /etc/certs/cert.pem
    privateKey: /etc/certs/private_key.pem
    caCertificates: /etc/certs/root.pem
```



https://github.com/tetratelabs/tetrate-todos

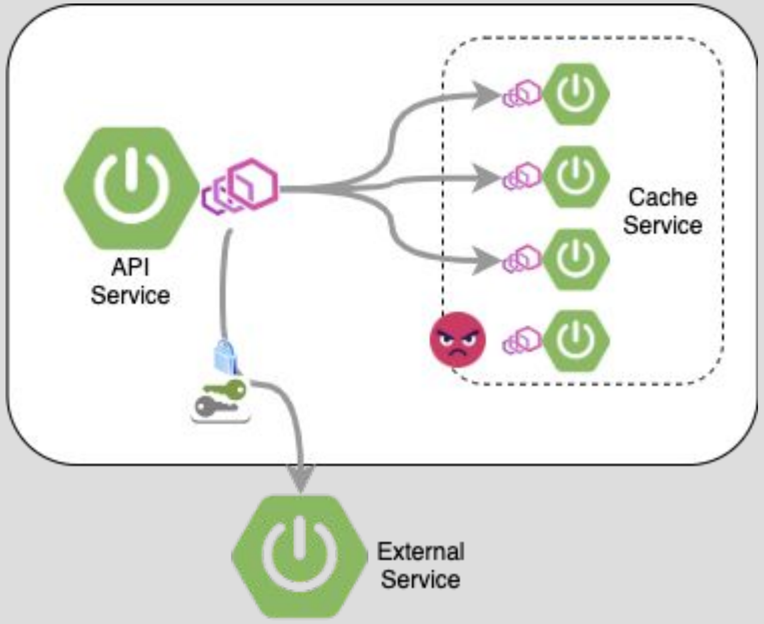
Resiliency

App Changes

- Remove Hystrix and/or Spring Cloud Circuit Breaker dependencies
- Remove @EnableCircuitBreaker, @HystrixCommand, @HystrixProperty, ect
- If implemented, remove any CircuitBreakerFactory impls.
- If using Reslience4j, remove config properties and annotations

https://github.com/tetratelabs/tetrate-todos

```
// Add in LB Strategy
host: todos-redis
trafficPolicy:
  connectionPool:
    tcp:
      maxConnections: 10
      connectTimeout: 1.0s
    http:
      http1MaxPendingRequests: 5
      http2MaxRequests: 50
  outlierDetection:
    consecutive5xxErrors: 10
    interval: 10s
    baseEjectionTime: 1m
```

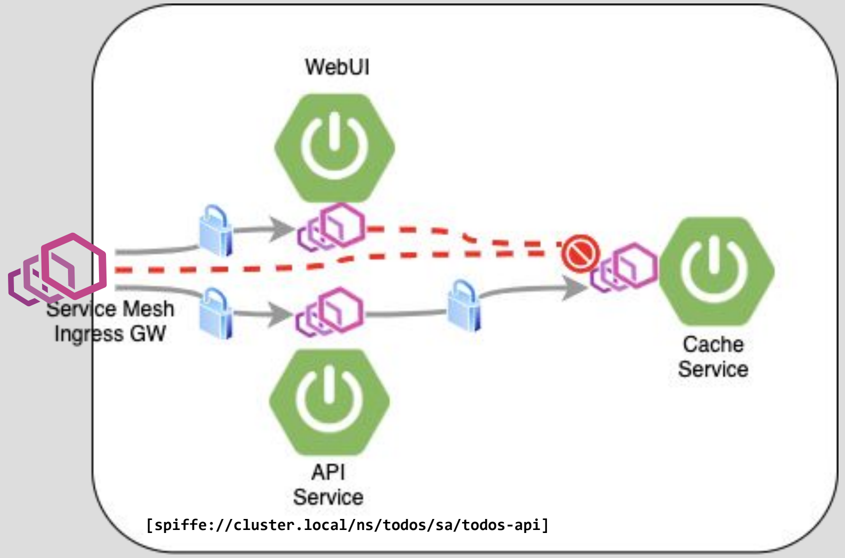


Security - Service to Service

App Changes

- Change L7 services to non-TLS
- Remove TrustStore/Keystore complexity from Jar/Container

```
// Add Service to Service AuthN Policy
metadata:
  namespace: todos
spec:
  mtls:
    mode: STRICT

// Add Service to Service AuthZ Policy
metadata:
  namespace: todos
spec:
  selector:
    matchLabels:
      app: todos-redis
 rules:
 - from:
   - source:
       principals: ["cluster.local/ns/todos/sa/todos-api"]
```



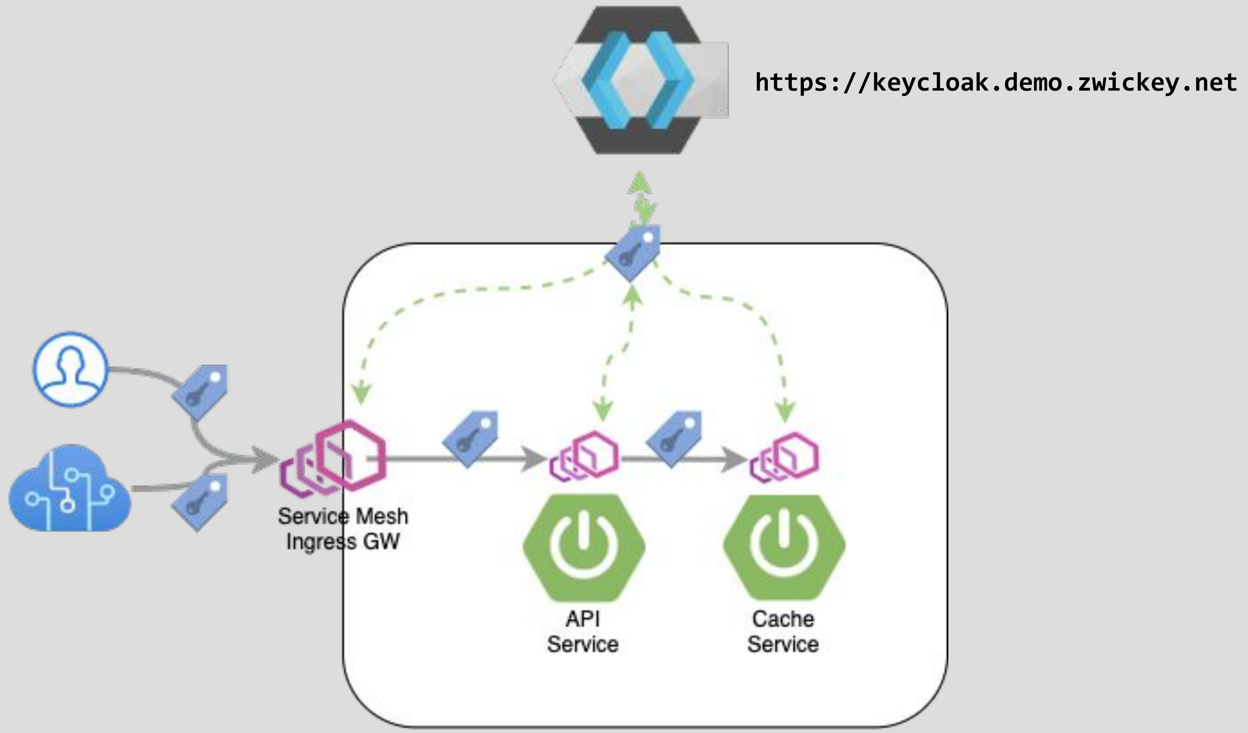
https://github.com/tetratelabs/tetrate-todos

Security - Request Level

App Changes

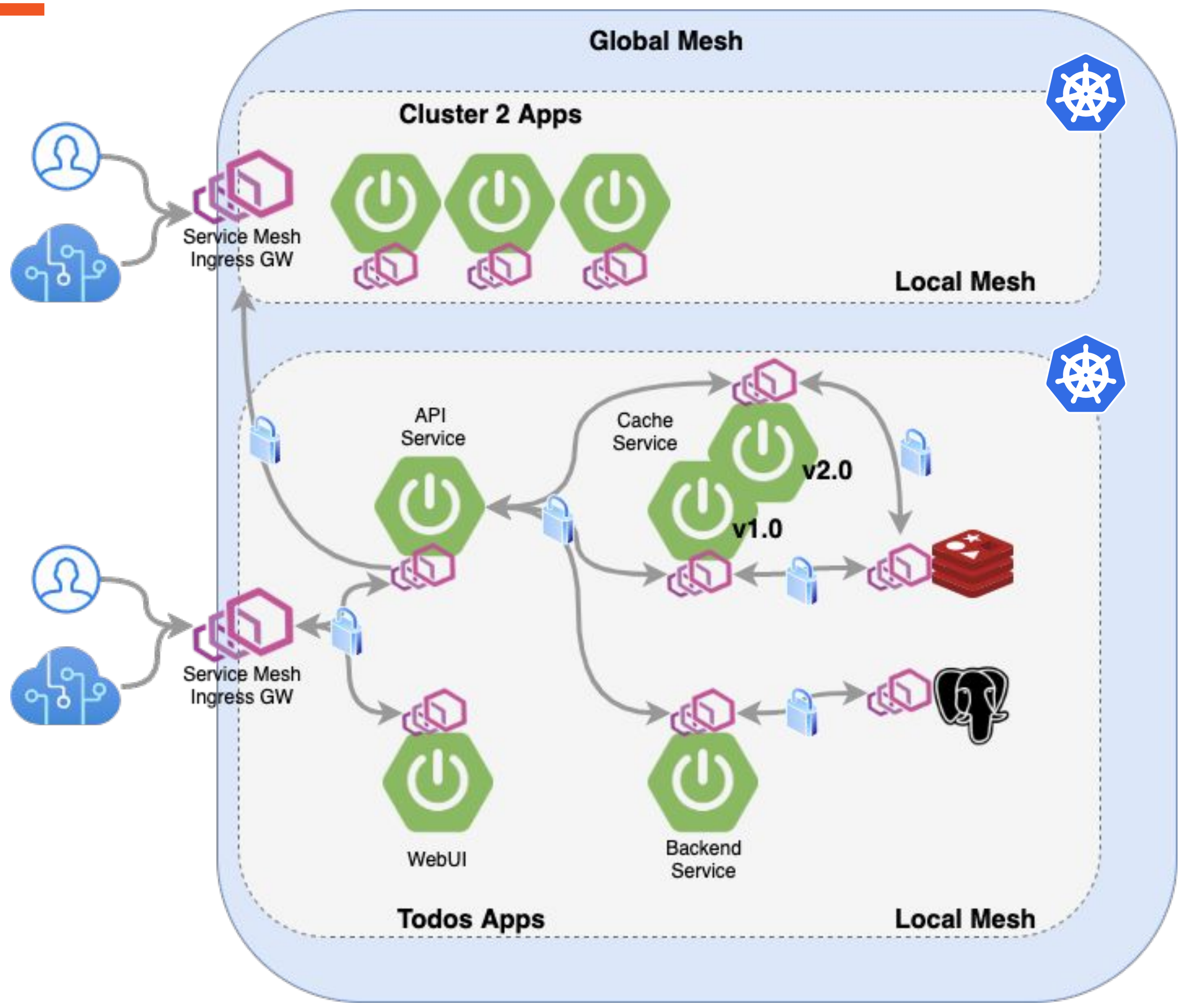
- ** Varies Widely **

```
// Add Request-Level Policy
rules:
 - from:
   - source:
       requestPrincipals: [ https://keycloak.demo.zwickey.net/auth/realms/tetrate/* ]
   to:
   - operation:
       paths:
       - /
   when:
   - key: request.auth.claims[roles]
     values:
     - todos-user-role
```



https://github.com/tetratelabs/tetrate-todos

Architecture with Service Mesh and Spring



https://github.com/tetratelabs/tetrate-todos

Wrap Up

- Decouples Application and Network
- Simplified App Architecture w/ CN Patterns Provided by Platform
- Unlocks and Simplifies Polyglot Approach
- Simplifies multi-cluster, multi-cloud, and non-cloud native apps
- More Robust Semantics for CD
- Consistent and Transparent Security Easily Added

tetrate

Thank You

Contact

@tetrateio | Tetrate | www.tetrate.io

For any further queries, feel free to contact us at info@tetrate.io